ASSIGNMENT 8

In this assignment you are going to create an experiment in which stimuli are presented at random positions on the screen. There might be some questions you have to answer in plain English (or Dutch). Put the answers to these questions in a *notepad* item which you place as the first item in your experiment. Call this notepad item *Answers*.

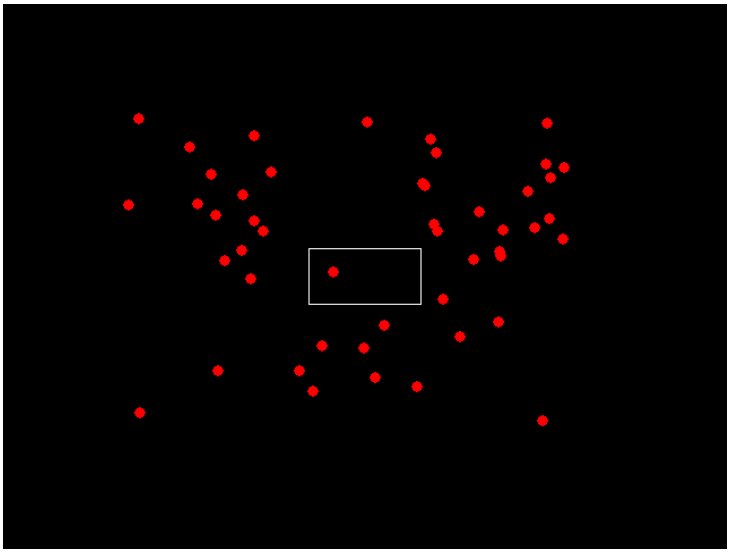
Once you are finished with this assignment, please save the file as

`Asgn_9_ <here your group number>.osexp`

(*for example* `Asgn_9_vanmoorselaar.osexp`)

before submitting it to canvas.vu.nl. Good luck!

Your task is to program a so-called "circle in a box" experiment. On each trial a rectangle is placed at the center of the screen and a series of red circles are presented sequentially at random positions. This continues until one of the circles (i.e., the center of the circle) is presented inside the rectangle (see Figure). The subject then has to press the spacebar as quickly as possible.



There are three conditions depending on the size of the circles. On one third of the trials the circles have a small size (radius 5 pixels); on another third of the trials the circles have a medium size (radius 10 pixels); on the final third of the trials the circles have a large size (radius 15 pixels).

The circles are always red and the center of the circles is always at least 100 pixels away from the edge of the display. The circle is considered in the box if the centre of the circle falls within the box.

The circles are not removed until the subject responds and the circles are presented sequentially at a randomly picked rate of 1 per 500, 750 or 1000 ms.

To make this experiment possible, we have to violate one of the golden rules that we have strictly followed during last assignments, namely: don't do any of the drawing in the *run* phase; do this in the prepare phase and only show a finished canvas in the run phase. For this assignment, you are going to draw (and show) stimuli during the *run* phase. We will have to show the canvas each time a red dot has been drawn and we don't know how many times we have to do this, because the red dots are randomly placed. The problem of preparing all canvases in the prepare phase is that we simply can't tell how long it takes before one of these random red dots ends up in the box of the middle of the screen and therefore we also have no idea how many canvases we would need to prepare in the *prepare* phase. Of course, it is possible to program this task and follow the strict prepare/run dichotomy, but this would make it unnecessarily difficult for you. Thus, to make your life as programmers a bit easier for once, you are only going to use one single canvas object and keep drawing to this object until a red dot appears in the center box. Then the drawing should stop and response measurements should start.

There is only one independent variable in this experiment, namely

- Circle size - 5, 10 or 15 px radius

The steps in a trial are (all in the run phase):

- Draw a white rectangle in the middle of the screen
- Draw a red dot at a random position on the screen. Keep doing this until the dot is drawn inside the rectangle
- Start measuring response times

General preparations

1) Try to write down the current programming task in *pseudo code* in the notepad item. You can use natural English (or Dutch!) sentences in pseudo code (thus sentences that are easy for you to understand), but try to formulate each line of your pseudocode in such a way, that it would be a small step to translate it into real Python code. In other words, it might be easiest to already think in terms of *if…else*, *while*, *for*, etc. Compare your pseudo code to that of class mates and see if they have reasoned differently than you.

2) Create a new experiment using the **default template** and place an instruction slide at the beginning and a goodbye slide at the end of the experiment. There should be one *experimental_loop* item (and thus no practice_loop), in which you present 2 blocks of 60 trials. Make sure the block_loop item contains three cycles representing the different circle radiuses (5,10 and 15px) and that the presentation order is *random*.

3) In the trial_sequence, add an inline_script (give it an appropriate name) followed by a keyboard_response item. Make sure that the only allowed response is the *spacebar*

4) A fixation dot should be presented for 1000 ms before the search display appears. This fixation dot should not be present on the dot display itself (i.e., the display with the square and dots).

The script

5) In the run phase of the inline script, import the modules you think you'll require, retrieve the desired variables from the loop items and create an empty canvas, which you reference with a variable.

6) Draw an unfilled square at the center of the screen. The square should be 200 pixels wide and high and its center coordinates should be the same as those of the screen. After the square has been drawn (and shown!) make sure it takes 1 second before the first dot appears.

7) While no circle has been drawn inside the rectangle, keep adding red circles with the same radius that is the value which was retrieved from block_loop to the display.

- The x,y coordinates can be chosen at random, with the restriction that the center of each circle is at least 100 pixels away from the edge of the display. A circle is considered in the box if at least half of the circle is within the box (thus if the center of the circle is on the outline of the box). Circles are allowed to overlap on the screen (that is, you do not need to check for overlap).
- Make sure you know what coordinate system you are using (i.e., what is the range of the x and the y axis)
- Each time you have drawn a circle you will have to also show the canvas, otherwise you will not see it appear. After a red circle has been shown, the next red circle should be drawn 500, 750 or 1000 ms later. This value can be randomly picked. If a red circle *did* appear inside the rectangle, no further red dots should be drawn and response measurements should start *with no delay*.

Note: You may notice that it's not possible to quit the experiment by pressing ESC before the first dot has been drawn inside the box. This is because OpenSesame only listens for ESC presses once it has arrived at the *keyboard_response* item. To make OpenSesame also listen to ESC presses before the first dot is drawn in the box, you will have to add the following code.

At the top of your script:

```
from openexp.keyboard import keyboard
kb = keyboard(exp)
```

Then, at the top of your while loop:

```
kb.get_key(timeout=0)
```

You don't need to understand what's going on here, because you'll probably have to do this trick in the current exercise only

Bonus question

8) We are going to make a little adjustment to the experiment. Adjust your code such that if a circle is presented on the top of the screen (i.e., above the horizontal midline) the next one must be presented below the midline, and vice versa. At the same time if a circle is presented on the left side of the screen (i.e., left of the vertical midline), the next one must be presented on the right, and vice versa. If a circle is presented exactly on the midline, either horizontal, or vertical, there are no restrictions for that plane. The exact location must still be selected at random.